C-CORE Publication No. 91-5

APPLYING ARTIFICIAL INTELLIGENCE TECHNIQUES FOR THE AUTOMATIC READING OF REMOTELY SENSED SYNOPTIC WEATHER DATA[1]

Charles Pang and Mona El-Tahan
Centre for Cold Ocean Resources Engineering (C-CORE)

Memorial University of Newfoundland
St. John's, Newfoundland
Canada A1B 3X5

ABSTRACT

In this paper, a new methodology for the automatic reading of handprinted numeric characters is applied to offshore operations in an attempt to enhance the entry of hand-recorded remotely sensed synoptic weather data into the computer. This new approach incorporates Artificial Intelligence techniques with algorithmic methods. The result is a 'trainable' heuristic number recognition system that can be configured to read weather data from log sheets (MANMAR, See Figure 1) prepared by Ice Observers and input them directly into the computer. Therefore, it can eliminate the time consuming task of manual data entry, reduces data entry errors and speeds up the transfer of weather information from offshore rigs to the shore-based Ice Room.

INTRODUCTION

An effective solution to the problem of automated character recognition is one that must involve intelligence or heuristics because visual recognition is a cognitive process. Therefore, it was decided to build a classification system that is based on techniques from artificial intelligence. Figure 2 shows the architecture of Handprinted Number Classification System (HNCS). The major components are the Skeletonization System (SkelSys), Heuristic Feature Extraction System (HFES) and the Heuristic Classification Engine (HCE). A MANMAR log sheet is first digitized by a Panasonic 35mm camera and stored in the standard UBC-Vision format. The handprinted numbers (symbols) on these log sheets are then isolated and stored in a table (10x10) for easy manipulation. Each symbol in the table is then fed into SkelSys one at a time. In SkelSys, each symbol is subjected to smoothing, skeletonization, noise pruning, segmentation into curve segments, polygonal approximation, and normalization into 20x25 pixels. Each normalized skeleton (represented by vertices) is then passed to the HFES. Select features (determined by the user) are then extracted and passed to the HCE. HCE then attempts to classify the symbol by matching its features with those in the Knowledge Base (KB).

HEURISTIC FEATURE EXTRACTION SYSTEM (HFES)

The feature extraction technique used in this system provides a powerful representational structure for the geometric and topological properties of each symbol. HFES combines both local and global information into each feature. Figure 3 shows the definitions of nine regions. The intersection of the top and left regions is the 'top-left' regions and so on. These regions will greatly enhance the spatial description of both local and global features. In essence, HFES provides the transformation from a skeleton (Figure 3) into sets of features (See Figure 4).

Each extracted feature is stored in a list called a Feature Vector (FV). The general structure of an FV is [feature-type, attribute_1, attribute_2, attribute_3, ... attribute_n]. The first element specifies the type of feature while the rest of the elements represent the attributes of that feature. For example, if the feature-type is an endpoint, the FV structure would look like: [endpoints, top, middle, bottom, left, centre, right]. A '1' in the place of any of the attributes (regions) in the list represents the existence of an endpoint in that particular region while a '0' means that there are no endpoints in that region. Therefore, using the first three attributes in the endpoint FV of Figure 4, it can be seen that there is one endpoint in the top, and two in the middle regions while the last three attributes indicate that there are two endpoints in the left region and one in the centre region. There are three endpoints in total. The interpretation of each FV varies depending on the type of features. The advantage of the FV representation scheme is that it is an economical and powerful way to store large amounts of descriptive and numeric information about the symbol. It also provides compatibility, in terms of data structures, between HFES

---

[1] Submitted to 14th Canadian Symposium On Remote Sensing, "Evolution Through Integration", 6-10 May 1991, Calgary.

that is coded in "C" and HCE which is coded in C-Prolog (Pereira, 1985). In HCE, the user selects the type of features to be used in the classification process and only those features that are selected will be extracted.

The curve detection and analysis procedure is used to locate any structural features such as C-curves, ⊃-curves, ∩-curves and U-curves that are present in the symbol. This is carried out by geometric analysis of the interior angles of each segment (polygon) and the overall orientation. Once a curve type has been determined, it is assigned to one of top, middle or bottom regions based on its proximity to the region. The symbol represented in Figure 4 has no C-curves or ∩-curves, but has one ⊃-curve in the top and one U-curve in the bottom regions.

Endpoint detection is fairly straight-forward. However, if two endpoints are close together, based on a predetermined threshold value (5 pixels), they will be merged. This is because characters can be malformed as a result of different writing styles and habits. Junction point detection is similar to endpoint detection. There are two types of junctions: T-junctions and X-junctions. A T-junction is defined as the meeting of three segments while an X-junction is a meeting of four segments. In a special case of T-junctions, two adjacent T-junctions are merged to form a single X-junction. This is to iron out irregularities that are sometimes created by SkelSys. A single thick X-junction is usually skeletonized to two T-junctions. Thus, merging recovers the original X-junction (Figure 5). The FV format for junctions is the same as that for endpoints (See Figure 3 and 4).

A loop is defined as a segment or a chain of segments whose starting point is the same as its endpoint. Sometimes, loops are imperfectly rendered and thus have open ends. If the gap between these open ends is small as in the case of endpoints, it will be closed. The FV for loops is [feature-type, top, #edges, middle, #edges, bottom, #edges]. Assignment of regions is also based on the loop's proximity to the regions (Figure 6). The #edges indicates the number of edges in the polygon that make up the loop. This information can be useful in determining how contorted the loop is. It was observed that some loops in certain classes of symbols are usually contorted in a similar way. This is still subject to further investigation.

Horizontal, Vertical, Slash and Backslash line segments are also features that are extracted by the HFES. The organizational information content of these line segments could be increased by specifying how the line feature is joined to the main body of the symbol. The criteria for determining these lines are given in Figure 7. The FV format for Vertical lines, Slash and Backslash is [feature-type, top, join, middle, join, bottom, join] while that for Horizontal line is [feature type, left, join, centre, join, right, join]. 'Join' represents the place where the line is connected to the main body. The possible values for 'join' are:

l - connected at the left	t - connected at the top
r - connected at the right	s - connected at both ends
b - connected at the bottom	x - not connected at all

HEURISTIC CLASSIFICATION ENGINE (HCE)

HCE is based on Minski's frame (Minski, 1975) representation scheme. HCE routines are categorized as, user-interface, classification, Knowledge Base (KB) access, Blackboard and Agenda routines (Marcus, 1986). In HCE, all the FV's from each symbol are combined into a frame. For example,

```
frame(8,   [[endpoints, 0, 0, 0,0, 0, 0],
            [x-junctions, 0, 1, 0, 0, 1, 0],
            [loops, 1, 6, 0, 0, 1, 6],
            [h_line, 0, o, 0, o, 0, o],
            [v_line, 0, 0, 0, o, 0, o] ]).
```

The name of this frame is '8' and its slots are the FV's, [endpoints, 0, 0, 0, 0, 0, 0], ..., [v_line, 0, o, 0, o, 0, o]. Slot values are the same as attribute-values with the exception of the first element which is the name of the slot. In HCE, there are routines to create and manipulate frames, and routines to perform static operations on the slots. The selection of features to be used in the classification is done via a user-interface HCE routine, menu.

There are two kinds of frames; Ordinary Frames (OF's) and Generalized Frames (GF's). OF's are frames that are

added to the KB without any processing. Frame "generalization" is a procedure in which all frames belonging to a same class will be examined to find similarities in them. A new frame is then created to encompass all these similarities. The new frame is called a Generalized Frame (GF). The theory behind frame generalization is that, if there is an overwhelming number of similarities among corresponding attribute-values for a certain feature-type, within a particular class of frames, it can be "generalized" that every member of this class should have that same attribute-value for that particular attribute-type. On the other hand, if there are no similarities then a generalization can also be made that the attribute-values are not good measures of the reliability of the attribute-type as a strong discriminant (i.e., the particular attribute-type is not significant to the class containment of this particular frame). As a result, an anonymous variable (indicated by an underscore in C-Prolog) is put in its place. This variable symbolizes a definite but unidentified object. The former generalization is called Type-1 generalization and the latter is called Type-2 generalization. If the system is unable to execute either forms of generalization, then the attribute-values are left as they are. An example of GF is shown below. Note that some of the slot values contain anonymous variables instead of actual values.

```
frame (3,   [[endpoints,1,_,1,2,_,0],
            [t_junctions,0,_,0,0,0,_],
            [ic-curve,1,0,_],
            [n-curve,0,0,0],
            [u-curve, 0,0,0], ...
            [h-lines,_,_,0,o,1,_] ]).
```

In HCE, the user can choose the threshold value (confidence limit) for the percentage of similarities that is needed in order for Type-1 generalization to occur. What is sought here is a *general* description to encompass all the members of a given class. Using this general description or concept, the classification process can then be expedited. Those frames that cannot be described in this *general* way become specific instances of this general concept and are called Ordinary Frames (OF's). When frame generalization is carried out for every attribute-type of every feature, the result is a more complex but more powerful and meaningful frame structure.

GF's serve as the backbone of the *inferential classification* strategy introduced in this research. For each class of symbol, there is one corresponding GF representing the general concept of the class. GF's provide an efficient and heuristic way to classify symbols. Using GF's, classification amounts to a search time that is only linear to the number of different classes of symbols. In HCE, as many as 78% of all successful classification is achieved using GF's. Since GF's were derived from generalizing OF's, some OF's will inevitably become redundant. These redundant OF's are eliminated. This is done by using GF's from each class and attempt to unify it with every OF within the same class. If unification is successful, then that OF will be removed from the KB. An interesting outcome is that as many as 60% of OF's are deleted from the KB (i.e., GF's are capable of representing as much as 60% of the frames in the KB). These redundancies are not due to character inputs that are similar since the frames were 'sifted' before they were added to the KB. They exist only as a result of Type-1 and Type-2 generalization. GF's are stored in a table format because of its small numbers while OF's are organized into a tree structure.

TRAINING

One of the primary objectives behind the development of HNCS is to create a system that is domain-independent (Garavaglia, 1987; Keller, 1987), that is, the system would be able to perform classification regardless of the knowledge with which it is furnished. Thus, the user has to feed domain knowledge into the system. This is known as training. In a training session, each symbol along with its FV's, will be displayed. The user will then decide if the symbol should be kept as part of the domain knowledge in the KB. Therefore, the user can carefully select only those symbols that best represent their class and keep them in the KB. In this way the KB would contain only very essential symbols, thus, optimizing memory usage and enhancing the accuracy of classification.

INFERENTIAL CLASSIFICATION

Inferential Classification is done by matching an input frame (IF) to frames in the KB. If the match is successful, the IF is classified as belonging to the same class as the frame with which the match succeeded. Matching is accomplished by the mechanisms of *unification* and *instantiation*. In inferential classification, unification is first

attempted between the IF (specific instance) and the GF's (general concept). The match is considered successful if and only if all the variables in a certain GF can be instantiated with the respective values in IF, and that all the other ground values in GF's can be successfully unified with the other ground values in IF. If the match fails, matching would then proceed to that with CF's (specific instances). This form of classification which is based on Type-1 and Type-2 generalization is believed to be the first of its kind and it introduces an entirely new perspective to the problem of developing generalized symbol recognizers.

CONCLUSION

HNCS was tested using 1100 symbols. Results show rates of 97.09% correct recognition, 1.18% rejection, 1.73% substitution error and a reliability of 98.25%. It has been noted (Mantas, 1986) that even the human eye which is undoubtedly the most well-trained recognition system in the world make some 4% mistakes when reading in the absence of context. In view of this fact, HNCS is practical and useful for the automatic reading of hand-recorded remotely sensed synoptic weather data in an offshore environment. Further research is currently being considered to improve the system for the recognition of the English Alphabet.

REFERENCES

Garavaglia, S. (1987), "Prolog, Programming Techniques And Applications", Harper & Row Publishers.

Keller, R. (1987), "Expert System Technology: Development and Application", Yourdon Press.

Mantas, J. (1986), "An Overview of Character Recognition Methodologies", Pattern Recognition 19, No. 6, PP 425-430.

Marcus, C. (1986), "Prolog Programming: Applications For Database Systems, Expert Systems And Natural Language Systems", Addison-Wesley.

Minski, M. (1975), "Framework For Representing Knowledge", in Patrick Henry Winston, Psychology of Computer Vision, McGraw-Hill Book Company.

Pereira, F. et al. (1985), "C-Prolog Users Manual", SRI International, Menlo Park, California.

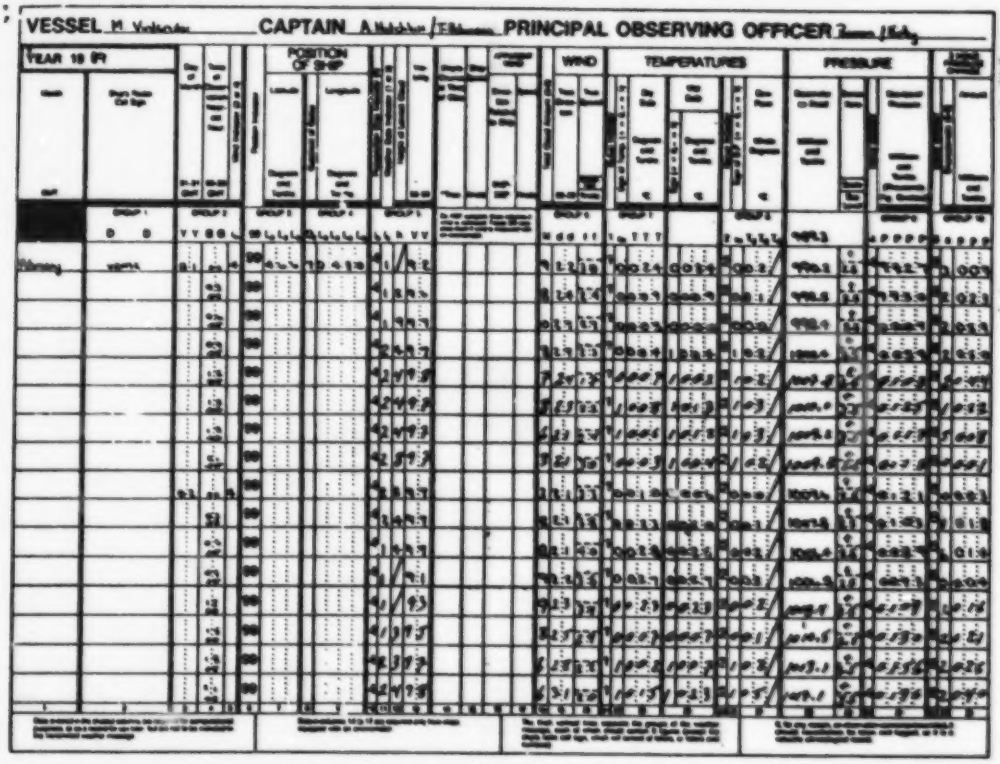

Figure 1. A MANMAR log Sheet

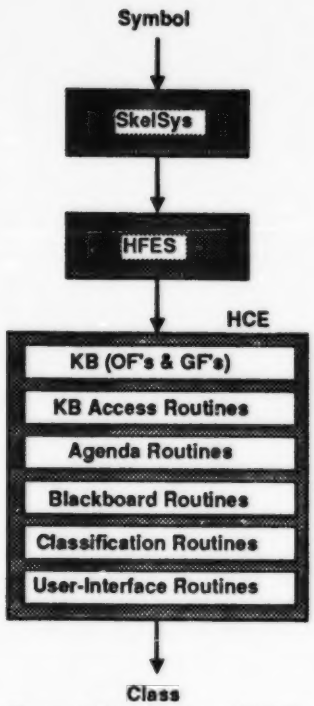

Figure 2. Architecture of HNCS

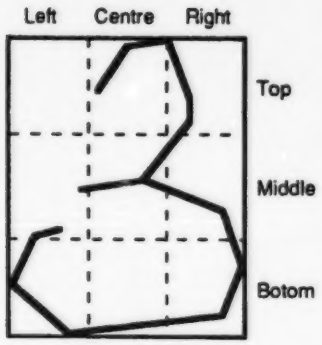

Figure 3. Symbol Regions

[endpoints,1,2,0,2,1,0]
[T-Jcnt,0,1,0,0,1,0]
[X-Jcnt,0,0,0,0,0,0]
[C,0,0,0]
[IC,1,0,0]
[U,0,0,1]
[N,0,0,0]
[Loops,0,0,0,0,0,0]
[H_line,0,0,0,0,1,s]
[V_line,0,0,0,0,0,0]
[Slash,0,0,0,0,0,0]
[BSlash,0,0,0,0,0,0]

Figure 4. FV's From Figure 3

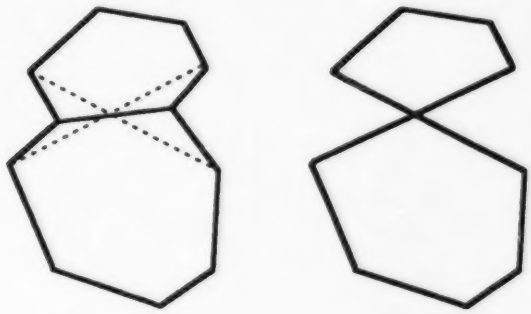

Figure 5. Effects of Merging Two T-junctions

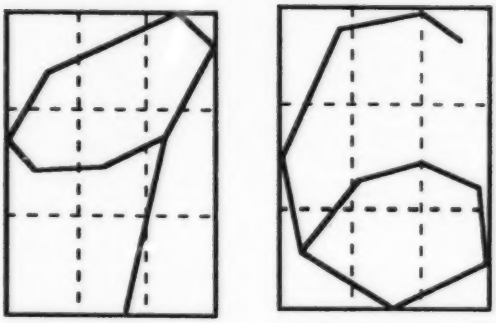

Figure 6. Assigning Loops To Regions

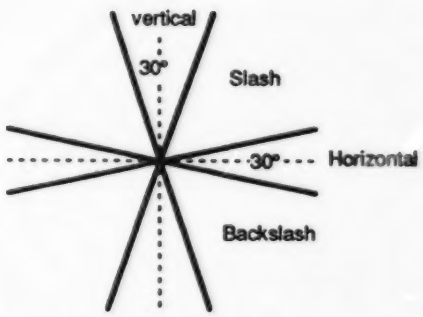

Figure 7. Determining Line Features